RESPIN++ – Deterministic Embedded Test

Lars Schäfer, Rainer Dorsch, Hans-Joachim Wunderlich
University of Stuttgart
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
{rainer.dorsch,wu}@informatik.uni-stuttgart.de

Abstract

RESPIN++ is a deterministic embedded test method tailored to system chips, which implement scan test at core level. The scan chains of one core of the system-on-a-chip are reused to decompress the patterns for another core. To implement the RESPIN++ test architecture only a few gates need to be added to the test wrapper. This will not affect the critical paths of the system. The RESPIN++ method reduces both test data volume and test application time up to one order of magnitude per core compared to storing compacted test patterns on the ATE. If several cores may be tested concurrently, test data volume and test application time for the complete system test may be reduced even further. This paper presents the RESPIN++ test architecture and a compression algorithm for the architecture.

1 Introduction

A system-on-a-chip (SoC) usually contains various predesigned and pre-validated embedded cores such as memory cores, processor cores, interface cores, mixed-signal cores and analog cores. This SoC design style increases the designer's productivity, but at the same time test complexity of the chip increases and the core's accessibility from the chip pins is reduced. The accessibility problem is addressed by isolating the cores during test from the system environment using test wrappers and transport the test data on a test access mechanism (TAM) from the test data source to the core under test (CUT) to the test data sink [19,20,23]. Nevertheless, higher test data volume is necessary to test the chip, higher ATE-chip bandwidth and more test pins are necessary to keep the test times low, and higher timing resolution is necessary to avoid yield loss. This leads to a significant rise in test costs as predicted in the ITRS road map [22].

This work was supported by the DFG under grant Wu 245/1-4.

Deterministic embedded test (DET) methods reduce the test data volume by storing a compactly encoded representation of the test patterns on the ATE and decoding this representation on-chip. Similarly, the test responses are encoded on-chip and compared with the encoded representation on the ATE. This paper focuses on test pattern compaction. In the next section we summarize prior work on on-chip decoders. Section 3 introduces the RESPIN++ test architecture and in Section 4 an encoding algorithm for the test data for the RESPIN++ architecture is presented. Some experimental results are presented in Section 5. Section 6 summarizes this work.

2 Prior Work

Storing test data in an encoded representation on the ATE was previously proposed many times. As on-chip decoders linear feedback shift registers (LFSRs) were proposed, which decode seeds for the LFSR to test patterns [11,12,17]. Other methods reduce the width of the test patterns and thus the test data volume to be stored on the ATE [3]. Many methods make use of the non-random nature of test patterns and use encoding methods known from data compression such as variants of Huffman coding [1,13,14], run-length coding [15] or Golomb coding [4,6]. If an embedded processor is available there are methods which only store differences between test vectors [16] or geometric shapes, which characterize the test vectors [9].

The recently presented RESPIN method (REusing Scan chains for test Pattern decompactIoN) [7,8] significantly reduces test data volume, test application time, or both compared to previously discussed decoding techniques. RESPIN uses the scan chains of one embedded core to decode test patterns for another core or interconnections. The RESPIN method consists of a test architecture and an algorithm which compacts the test data for the architecture. Figure 1 illustrates the test architecture of RESPIN in an example system. The scan chains of the embedded tester core (ETC), which is the MPEG core in this example, are used for the decompression of the patterns for the CUT, which

is the CPU core in this example. For each core used to decode test patterns only an additional multiplexer and some wiring is needed in the test wrapper. The critical paths of the system remain untouched.

In the RESPIN architecture the compressed test information is transported from the ATE to the chip on a $k = 1$-bit *narrow test access mechanism (TAM)*. To transform the encoded test data into test patterns the scan chains of the ETC are reused. The test patterns are transported on a l-bit *wide TAM* with a high bandwidth from the ETC to the CUT (see Fig. 1).

For manufacturing test, test application time is often more important than test data volume, especially if the ATE memory is not fully utilized by the encoded test data. RESPIN++ allows to trade off efficiently between test data volume and test application time, which is impossible in the RESPIN architecture. The only change with respect to the RESPIN architecture is the addition of XOR gates to the inputs of the ETC's scan chains. Thus this architecture is called RESPIN+$\oplus$ or just RESPIN++. The test data encoding algorithm is entirely new.

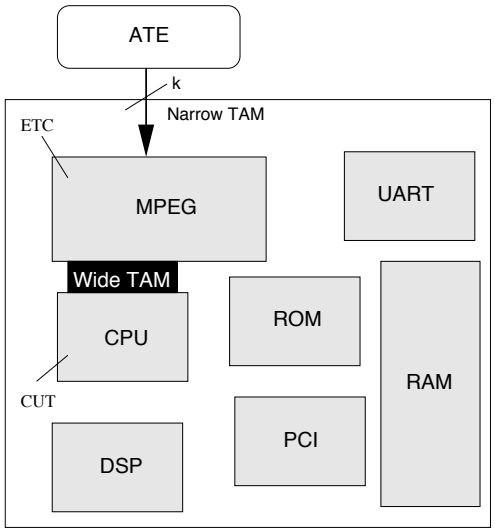
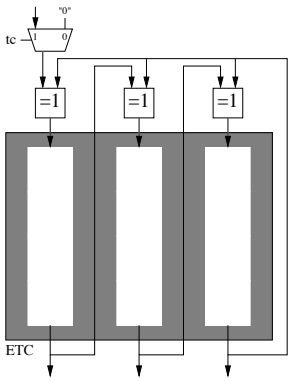


Figure 1. SoC with the RESPIN Test Architecture

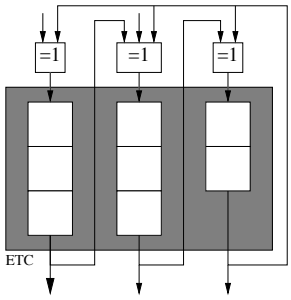
3 RESPIN++ Test Architecture

The RESPIN and the RESPIN++ test architectures use scan chains of an arbitrary core to decode the test data stored on the ATE. The core reused for decoding is called *embedded tester core* (ETC). The core which is tested is called *core under test* (CUT).

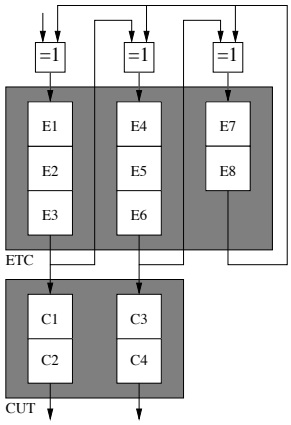
In the RESPIN++ test architecture either k bit per clock cycle are scanned in from the narrow TAM or k bit are



(a) ETC Test Architecture



(b) Modification for 2 Bits per Clock Cycle



(c) Complete Architecture

Figure 2. RESPIN++ Test Architecture

scanned in every n-th cycle in order to facilitate low cost testers operating at a larger period n. The selection of the narrow TAM width k and tester period n allow a trade-off between test application time and test data volume. E.g. a smaller period n and a larger narrow TAM width k leads to lower test application time and increases test data volume for a given core.

Both the CUT and the ETC are equipped with scan chains and isolated from their environments by a test wrapper during test. In order to operate the ETC as decoder in the RESPIN++ architecture it is necessary to modify its test wrapper slightly. To the input of each scan chain an XOR gate is added. Figure 2(a) shows an ETC configuration for $k = 1$, Figure 2(b) for $k = 2$. The inputs of the XOR gates are connected to

- the output of the adjacent scan chain, as in the serial access mode,
- a feedback wire from the output a scan chain in serial access mode, and
- the narrow TAM (only the first k scan chains).

If the ETC is supposed to operate at a frequency higher than the ATE's frequency multiplexers with a control input t_c may be added to narrow TAM. t_c may be used to force these inputs of the XOR gates to 0, which are connected to the narrow TAM. The signal t_c selects between *feedback mode* ($t_c = 0$) and *scan mode* ($t_c = 1$). Any ratio of cycles between the two modes is possible. The parallel test access outputs of the ETC are connected to the parallel test access inputs of the CUT via the wide TAM (see Fig. 2(c)).

The RESPIN++ method achieves at least the same fault coverage as compacted test patterns, if the simple design rules of the RESPIN architecture are followed, which avoid that the content of a scan cell in the ETC is mapped to two scan cells of the CUT [7].

4 Encoding Algorithm

The pattern scanned into the CUT depends on the content of the scan chains of the ETC, which is the state of the ETC. The state of the ETC is described by a state vector $\mathbf{s}$. In each clock cycle in scan or feedback mode, a state transition occurs in the ETC, which is described by a matrix multiplication. This leads to a sequence of states $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \ldots$ which controls the test patterns scanned into the CUT. In feedback mode the next state may be obtained by multiplying the current state with the feedback matrix $\mathbf{R}$:

$$\mathbf{s}_{i+1} = \mathbf{R} \cdot \mathbf{s}_i. \tag{1}$$

The following feedback matrix describes the example from Figure 2(c).:

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} \text{E1} \\ \text{E2} \\ \text{E3} \\ \text{E4} \\ \text{E5} \\ \text{E6} \\ \text{E7} \\ \text{E8} \end{pmatrix}$$

In scan mode the next state can be derived from the scan matrix $\mathbf{S}$ and the ATE vector $\mathbf{v}$:

$$\mathbf{s}_{i+1} = \mathbf{R} \cdot \mathbf{s}_i. + \mathbf{v} \tag{2}$$

$$\mathbf{v} = \begin{pmatrix} x_i \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

When scanning in two bits per clock cycle (see Figure 2(b)) feedback matrix and ATE vector look slightly different and are denoted as $\mathbf{R}_2$ and $\mathbf{v}_2$ respectively:

$$\mathbf{R}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{v}_2 = \begin{pmatrix} x_i \\ 0 \\ 0 \\ x_j \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The ATE vector introduces variables into the state vector $\mathbf{s}$. For each test vector $\mathbf{t}_i \in T$ it may be checked, if the variables in the state vector $\mathbf{s}$ may be set in such a way that the appropriate rows of $\mathbf{s}$ satisfy the specified bits in $\mathbf{t}_i$. This may be done by solving the resulting set of linear equations. To encode all test patterns out of T the algorithm repeatedly passes through the following three phases:

1. In the *scan phase m* new variables are scanned in (see Equation 2).
2. In the *feedback phase* the ETC is operated n cycles in feedback mode. This fills up the pattern and does not require data from outside. (see Equation 1).
3. In the *encoding phase* as many test vectors as possible are encoded by solving the set of linear equations.

If the set of linear equations has a unique solution for a test vector $t_i \in \mathbf{T}$ then t_i is removed from $\mathbf{T}$ and the variables present in the set for linear equations are set appropriately. If more than one solution exists for the set of linear equations, the operations are used as constraints for all further sets of equations.

The phases 1 through 3 are repeated until all vectors of the test set T are encoded.

4.1 Restriction of Search Space

In every scan phase new variables are added to the state vector. This may result in a continuous growth of the number of variables in the state vector. Only in the rare case, when there exists a unique solution for the set of linear equations in the encoding phase, variables are fixed. In order to limit the continuous growth of the number of variables, some variables are forced to values, which satisfy all constraints, if the number of unset variables x_i exceeds l. Setting only some variables improved encoding quality and distinguishes the encoding algorithm from traditional LFSR reseeding methods, which set all variables, when linear equations are solved [17,18,21]. To guarantee this limit, in state $\mathbf{s}_i$ only the variables $x_{i-l} \ldots x_{i-1}$ ($\forall i \geq l$) are still unset. All variables x_j with $j \leq (i-l)$ have been set to a fixed value in the encoding phase of a previous cycle.

Additionally the probability for encoding of test patterns depends on the number of constraints introduced during the encoding of previous test patterns. If the encoding probability is very low, the number of unsuccessful attempts to encode a test pattern per state is limited to a user defined number u.

4.2 Example

This example shows how to compact the deterministic test set $\mathbf{T} = \{\mathbf{t}_1 = (10-1), \mathbf{t}_2 = (-011), \mathbf{t}_3 = (11-0)\}$ using the test architecture from Figure 2 (b). The test set $\mathbf{T}$ consists of three partially specified 4-bit test cubes. The hyphen indicates a don't care bit. The first bit of a test cube $\mathbf{t}_i$ is scanned into the first scan cell C1 of the CUT, the second into scan cell C2, and so on. Let $l = 11$ variables to be maintained in the state vector of the ETC, let the ETC operate $m = 2$ cycles per pattern in the scan phase and $o = 1$ cycle in the feedback phase. In the example, in each cycle in the scan phase a new variable is scanned in.

4.2.1 Encoding

At first the circuit is operated in scan mode until all 11 variables are scanned in (see Fig. 3 (a) and (b)). Next encoding $\mathbf{t}_1$ with the state of the CUT leads to the following set of linear equations :

$$\begin{aligned} x_8 &= 1 \\ x_7 &= 0 \\ x_4 &= 1 \end{aligned} \tag{3}$$

This set of equations is satisfiable. Hence $\mathbf{t}_1$ is removed from $\mathbf{T}$ and the set of equations is saved as a constraint for encoding the remaining patterns. Now $\mathbf{t}_2$ is encoded with the same state of the CUT. With the constraints from above (equation 3) this yields the following set of equations

$$\begin{aligned} x_8 &= 1 \\ x_7 &= 0 \\ x_4 &= 1 \\ x_5 &= 1 \end{aligned} \tag{4}$$

This set of equations is also satisfiable. Therefore $\mathbf{t}_2$ is removed from $\mathbf{T}$ as well and the set of equations (4) is the new set of constraints. It is impossible to encode $\mathbf{t}_3$ in this step. Next, the two oldest variables x_1 and x_2 are set. Because they do not appear in the set of constraints (4) they are set to an arbitrary value. In this example they are set to 0.

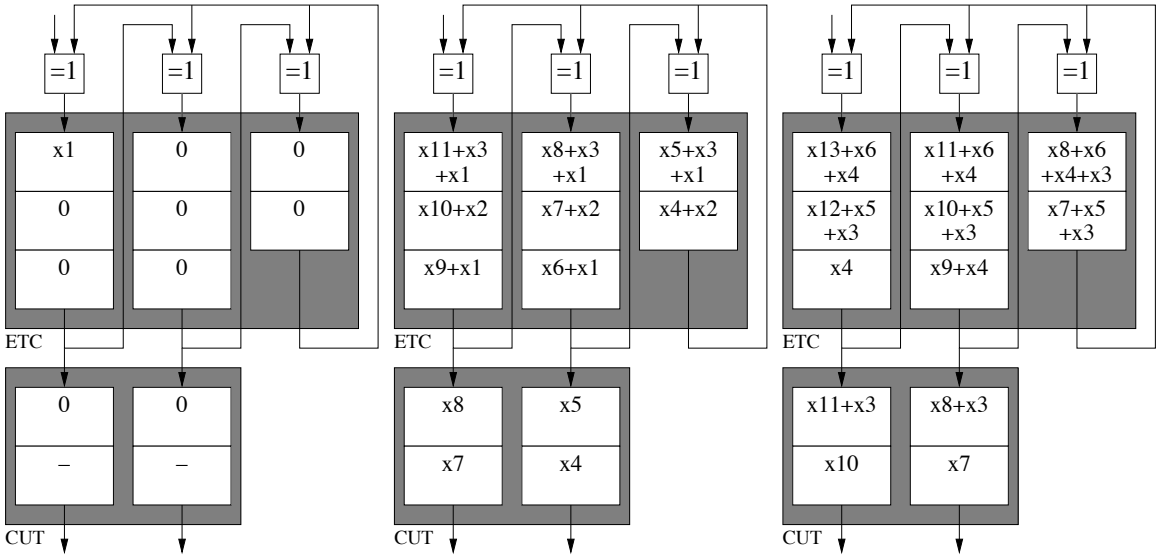
The ETC is afterwards operated one cycle in feedback mode. After that (see Fig. 3 (c)) it is operated two cycles in scan mode. Then the next encoding phase follows, trying to match the remaining test cube $\mathbf{t}_3$. The set of constraints (4) must still be valid, so the resulting set of equations is:

$$x_8 = 1 \tag{5}$$
$$x_7 = 0 \tag{6}$$
$$x_4 = 1 \tag{7}$$
$$x_5 = 1 \tag{8}$$
$$x_{11} + x_3 = 1 \tag{9}$$
$$x_{10} = 1 \tag{10}$$
$$x_8 + x_3 = - \tag{11}$$
$$x_7 = 0 \tag{12}$$

This set of equations is satisfiable as well, hence all test vectors are compressed and all remaining variables $x_3 \ldots x_{13}$ are set according to the set of equations (5) to (12). Variables that are not specified in those equations are set arbitrarily.

4.2.2 Test Data Volume and Test Application Time

Due to the fact that this example is small, no reduction in test data volume can be observed. It takes 13 bits from the ATE ($x_1, \ldots, x_{13}$) to apply the three test cubes which contain only 12 bits. Note that $x_1$and x_2 were set arbitrarily and $x_3, x_6, x_9, x_{12}, x_{13}$ remain unspecified. These unspecified bits can be used to encode more test cubes. In real application only a very small fraction of bits remains unspecified. The



(a) after 1 cycle (b) after 11 cycles (c) after 14 cycles

Figure 3. Scan Phase

total test data volume reduction is in this example negative $\frac{12-13}{12} = -8.3\%$.

Test application time is computed as follows: It takes 11 cycles in scan mode to fill ETC and CUT, then 1 cycle in feedback mode and another 2 cycles in scan mode. This sums up to 14 clock cycles.

4.3 Algorithmic Description

The formal representation of the algorithm used for test data compression in the RESPIN++ architecture is shown in Algorithm 1. After an initialization sequence, the three phases described before are repeated until all test patterns in **T** are encoded.

Algorithm 2 shows the encoding phase in more detail. The patterns are encoded iteratively, the algorithm stops if the encoding of u patterns failed. To make sure, that only l variables are in the set of linear equations, the oldest variables are set in a way that they are compatible to the constraints.

Algorithm 1 RESPIN++ Algorithm

```
TestSet T;
TestCube t ;
INT l; //variables maintained in state vector
INT u; //max unsuccessful attempts
INT m; //new variables per pattern
INT o; //cycles in feedback phase
ScanPhase(k − m);
while T ≠ ∅ do
   ScanPhase(m);
   FeedbackPhase(n);
   EncodingPhase();
end while
```

5 Experimental Results

Experiments were performed with SoCs which contain the ISCAS89 benchmarks [2] as cores. For the CUT it was assumed that each scan chain had a length of 100 bit each. The last chain was shorter, its length depends on the total number of scan cells in the CUT. The chains of the ETC contained 101 scan cells each. A significant influence of the choice of the chain lengths on the results was not observed.

The algorithm associated with the RESPIN++ test architecture was implemented in C++. The sets of linear boolean equations were solved using Gauss-Jordan elimination. The number of unsuccessful attempts was limited to $u = 100$ per state. The deterministic test patterns were generated by a commercial ATPG tool. The ATPG patterns contained the bit values 0, 1, and "don't care". The ATPG patterns were encoded by RESPIN++. To reduce test data volume a two level technique was used. In the first pass, only test vectors for faults, which were not detected by 400 random patterns, were encoded. In the second pass test vectors for all re-

Bench-	Compacted Test Set			RESPIN++				
mark	$\|TS_C\|$	Bits	Cycles	Bits		Patterns	Cycles	
				#	Red.	#	#	Red.
s5378	117	25,038	25,252	17,332	31 %	165	17,497	31 %
s9234.1	145	35,815	36,062	17,198	52 %	161	17,359	52 %
s13207.1	240	168,000	168,700	26,004	85 %	242	26,246	84 %
s15850.1	119	72,709	73,320	32,226	56 %	306	32,532	56 %
s38417	93	154,752	156,416	89,132	42 %	854	89,986	42 %
s38584.1	131	191,784	193,248	63,232	67 %	599	63,831	67 %

Table 1. Comparison with compacted test set

Algorithm 2 EncodingPhase()

```
INT i = 0; //unsuccessful attempts
for all t ∈ T do
  encode t with the current state;
  solve resulting set of equations fulfilling all constraints;
  if satisfiable then
    T = T \ {t};
    add this set of equations to the constraints;
  else
    i = i + 1;
  end if
  if i ≥ u then
    solve constraints;
    if several solutions then
      set oldest m variables;
    else
      set all l variables;
      /* there is at least one solution because only solvable sets of equations are added to the constraints */
    end if
    break;
  end if
end for
```

maining faults were encoded. These remaining faults were the faults not detected by the deterministic patterns of the first pass and by link patterns that occur, when encoding the deterministic patterns in the first pass. For all benchmarks 100% fault efficiency was achieved.

5.1 Comparison with Compacted Test Patterns

This section compares test application time and test data volume of the RESPIN++ architecture and a standard scan test architecture with unencoded but compacted test patterns stored on the ATE. It is assumed that both architectures may use 1 ATE scan channel for testing the CUT. In the RESPIN++ architecture a $k = 1$ bit narrow TAM was chosen and the first $m = 100$ bits per test pattern were scanned into the ETC from the narrow TAM (i.e. one cycle in feedback mode per test pattern). In the standard architecture the test patterns are scanned into the CUT in serial mode.

Table 1 shows the results of the comparison. Column 1 contains the name of the benchmark, Column 2 the number of test patterns in the compacted test set, Column 3 the number of bits to be stored on the ATE and Column 4 the number of cycles needed to test the circuit. Comparing this with the test data volume (Column 5) and test application time (Column 8) of the RESPIN++ architecture shows a reduction of up to 85% for both values. The comparison in the number of patterns applied to the CUT (Column 2 and 7) shows that for the large benchmarks, the RESPIN++ method needs more patterns. Scanning in more than one bit per cycle from the ATE (i.e. k $>$ 1) will reduce the number of applied patterns in the RESPIN++ method considerably. This implies that for large designs the feedback phase is not necessary in manufacturing test.

5.2 Comparison with DET Methods

In this section the RESPIN++ method is compared with other DET methods, which decode test data on the chip. Two methods, *FDR* and *Golomb* implement the decoding algorithm in hardware [4,5], another method, *GPBC*, needs a microprocessor to decode the test data [9]. Comparing the test data volume of all four methods (columns 2, 4, 5 and 6 of Table 2) shows that although the RESPIN++ method does not need a microprocessor for test pattern decoding, the test data volume is only slightly larger than the test data volume of GBPC. FDR and Golomb require often a higher test data volume than RESPIN++.

The test application time is given by a few initialization cycles for the ETC plus the test data volume applied to the CUT in bits divided by the throughput of the decoder (=number of bits generated by the decoder in each clock cycle). A low test application time therefore requires a high throughput of the decoder. GBPC is limited by the width of

Bench-	FDR	GPBC	Golomb	RESPIN++
	[5]	[9]	[4]	
mark	Bits	Bits	Bits	Bits
s5378	12,307	10,057	-	17,332
s9234.1	21,647	14,666	22,495	17,198
s13207.1	35,236	22,458	35,122	26,004
s15850.1	36,284	24,446	30,581	32,226
s38417	74,905	60,478	91,088	89,132
s38584.1	93,878	-	120,350	63,232

Table 2. Compressed Test Data

the data path of the CPU (typically up to 32 bits). Golomb and FDR are variable-to-variable codes, i.e. the throughput is variable. The RESPIN++ architecture generates as many bits per clock cycle as scan chains of the ETC are available. In practice, several hundred scan chains are often present in a core [10], thus the RESPIN++ architecture has a high and constant throughput of the decoder.

5.3 System Test

This sections shows the control of the user over the optimization goal test application time and test data volume. We performed experiments with $k = 1$ and varied the number of bits per test patterns scanned in from the narrow TAM. For larger cores the width k of the narrow TAM may be varied and the ETC obtains in each cycle data from the narrow TAM. We discuss the implications of this feature on system test.

During system test, all cores of the system must be tested. The system integrator may choose to use the bandwidth of the narrow TAM to transport the compressed test data of one core to the ETC of this core and if the test of this core is completed another core is tested in the same way. Alternatively, several cores may be tested concurrently and the narrow TAM may be time multiplexed to transport test data to the ETCs of the cores tested concurrently. Testing cores concurrently requires more ETCs but it reduces the test application time and test data volume on the ATE for the complete system test considerably.

In the previous experiments the cores had scan chains of length 100 and 100 bits were scanned in the ETC per test pattern. If two cores would be tested concurrently, in the first 50 cycles, the data are for the ETC which tests the first core and the second 50 cycles for the ETC which tests the second core. In the RESPIN++ method this is modeled by operating the ETC 50 cycles in scan phase and 50 cycles in feedback phase.

To assess the effect of concurrent test of the cores on the test data volume and test application time for the system, an experiment was performed in which we had assumed that a system had 40 cores of the type s9234. Then we performed a system test, in which we varied the number of cores tested concurrently. We did not make use of the fact that the cores in the experiment were identical.

The results of the experiment in table 3 show that the more cores are tested concurrently, the lower becomes the system test application time and system test data volume. Column 1 shows the number bits per test pattern which are loaded from the ATE to the ETC. If m bits per test pattern are loaded from the ATE, then the encoding took into account that the ETC operated m clock cycles in scan and $101 - m$ clock cycles in feedback mode. Column 2 shows the resulting number of bits which need to be stored on the ATE per core. Column 3 shows number of patterns applied to test this core and Column 4 the test application time per core. Columns 5 to 7 refer to the system test. Column 5 gives the maximum number of cores which may be tested concurrently using a single bit time multiplexed narrow TAM. The total test data volume in Column 6 is given by the sum of test data volume of the individual cores, in the example by 40 times Column 2. The test application time in Column 7 for the system test is given by the sum of the test application time of the cores divided by the number of cores tested concurrently. When 20 cores are tested concurrently, the test data volume and the test application time for the complete system are reduced by approximately 60% compared to a sequential test of all cores in the RESPIN architecture.

6 Summary

The RESPIN++ test method has been developed for the deterministic embedded test of SoCs. This deterministic embedded test method consists of a test architecture and an encoding algorithm for this architecture. In the architecture test patterns for a core are decoded with the scan chains for another core, which is idle during the test of the first core. In order to reuse the scan chains of a core as an embedded tester core, only minor modifications in its test wrapper are necessary. The associated encoding algorithm is based on solving sets of linear equations and on multiplying matrices. The method is characterized by a high throughput of the on-chip test data decoder. The method reduces test application time and test data volume up to one order of magnitude per core and even more, if cores may be tested concurrently in the system test.

References

[1] I. Bayraktaroglu and A. Orailoglu. Test Volume and Application Time Reduction Through Scan Chain Concealment. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 151–155, 2001.

Bits	Core			System		
per Pattern	Bits	Patterns	Cycles	concurrency	Bits	Cycles
100	17,198	161	17,359	1	687,920	694,360
50	12,048	223	23,421	2	481,920	486,400
25	9,048	318	33,216	4	361,920	365,120
10	7,958	706	72,204	10	318,320	321,160
5	6,988	1,218	123,916	20	279,920	281,840

Table 3. System Test

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinatorial Profiles of Sequential Benchmark Circuits. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, pages 1229–1234. IEEE, 1989.

[3] K. Chakrabarty and B. T. Murray. Design of built-in test generator circuits using width compression. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:1044–1051, October 1998.

[4] A. Chandra and K. Chakrabarty. Efficient test data compression and decompression for system-on-a-chip using internal scan chains and Golomb coding. In *Proceedings of the Design Automation and Test in Europe (DATE)*, pages 145–149, Munich, March 2001. IEEE Computer Society Press.

[5] A. Chandra and K. Chakrabarty. Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 42–47, 2001.

[6] A. Chandra and K. Chakrabarty. System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20, March 2001.

[7] R. Dorsch and H.-J. Wunderlich. Reusing scan chains for test pattern decompression. In *Proceedings of the IEEE European Test Workshop (ETW)*, pages 124–132, Stockholm, Sweden, May 2001. IEEE Computer Society Press.

[8] R. Dorsch and H.-J. Wunderlich. Tailoring ATPG for Embedded Testing. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 530–537, Baltimore, MD, October 2001. IEEE Computer Society Press.

[9] A. El-Maleh, S. al Zahir, and E. Kahn. A Geometric-Primitives-Based Compression Scheme for Testing Systems-on-Chip. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 54–59. IEEE Computer Society Press, 2001.

[10] X. Gu, S. Chung, F. Tsang, J. A. Tofte, and H. Rahmanian. An Effort-Minimized Logic BIST Implementation Method. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 1002–1010. IEEE Computer Society Press, 2001.

[11] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois. Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers. *IEEE Transactions on Computers*, 44(2):223–233, February 1995.

[12] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois. Generation of Vector Patterns through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 120–129, Washington, DC, 1992. IEEE, IEEE Computer Society Press.

[13] V. Iyengar, K. Chakrabarty, and B. Murray. Deterministic Built-in Pattern Generation for Sequential Circuits. *Journal of Electronic Testing Theory and Applications (JETTA)*, 15(1/2):97–114, August/October 1999.

[14] A. Jas, J. Ghosh-Dastidar, and N. Touba. Scan Vector Compression/Decompression Using Statistical Coding. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 114–120, Dana Point, CA, 1999. IEEE Computer Society Press.

[15] A. Jas and N. Touba. Test Vector Decompression Via Cyclical Scan Chains and Its Application to Testing Core-Based Designs. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 458–464, Washington, DC, 1998. IEEE Computer Society Press.

[16] A. Jas and N. Touba. Using an Embedded Processor for Efficient Deterministic Testing of Systems-on-a-Chip. In *International Conference on Computer Design (ICCD)*, pages 418–423, 1999.

[17] B. Koenemann. LFSR-Coded Test Patterns for Scan Design. In *Proceedings of the European Test Conference (ETC)*, pages 237–242, München, 1991.

[18] C. V. Krishna, A. Jas, and N. A. Touba. Test Vector Encoding Using Partial LFSR Reseeding. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 885–893, 2001.

[19] E. J. Marinissen. On Using IEEE P1500 SECT for Test Plug-n-Play. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 770–777, Atlantic City, NJ, 2000. IEEE, IEEE.

[20] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel. Towards a Standard for Embedded Core Test: An Example. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 616–627. IEEE, 1999.

[21] J. Rajski, J. Tyszer, and N. Zacharia. Test Data Decompression for Multiple Scan Designs with Boundary Scan. *IEEE Transactions on Computers*, 47(11):1188–1200, November 1998.

[22] Semiconductor Industry Association. *The International Technology Roadmap for Semiconductors (ITRS)*. International SEMATECH, Austin, TX, 2001.

[23] Y. Zorian. Test Requirements for Embedded Core-based Systems and IEEE P1500. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 191–199, 1997.